VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014

Tomáš Weiss

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Weiss**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: RAYNET s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Krömer, Ph.D.**

Konzultant bakalářské práce: Ing. Aleš Seifert

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 5. května 2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2014 .....

Rád bych poděkoval Vysoké škole báňské – Technické univerzitě v Ostravě za umožnění absolvování bakalářské práce formou odborné praxe. Mé upřímné poděkování také náleží celému kolektivu společnosti RAYNET s.r.o. za obrovskou podporu a nespočet předaných praktických zkušeností. Děkuji také Ing. Pavlu Krömerovi, Ph.D. za poskytnutí důležitých informací a připomínek bez kterých by tato práce nemohla vzniknout.

Abstrakt

Práce má za cíl popsat průběh a rozsah absolvování odborné praxe ve společnosti RAYNET s.r.o. Součásti je popis implementace zadaných úkolu v průběhu odborné praxe. Hlavní důraz je kladen na rozbor návrhu architektury a zvolených technologií pro webový server RAYNET Cloud CRM.

Klíčová slova: Cloud CRM, RAYNET, webový server

Abstract

The work aims to describe the progress that has been achieved during completion of a professional practice in RAYNET s.r.o. company. It also includes the implementation description of given tasks. The main emphasis is on analysis of architecture design and chosen technology as a solid base for RAYNET Cloud CRM web server.

Keywords: Cloud CRM, RAYNET, web server

Seznam použitých zkratek a symbolů

ACL	–	Access Control List
AJAX	–	Asynchronous JavaScript and XML
AMQP	–	Advanced Message Queuing Protocol
API	–	Application Programming Interface
AWS	–	Amazon Web Services
CGI	–	Common Gateway Interface
CRM	–	Customer Relationship Management
CSP	–	Central Service Provider
EC2	–	Elastic Compute Cloud
GUI	–	Graphical User Interface
HTTP	–	Hypertext Transfer Protocol
IDE	–	Integrated Development Environment
IP	–	Internet Protocol
JSON	–	JavaScript Object Notation
MVC	–	Model View Controller
REST	–	Representational State Transfer
RPC	–	Remote Procedure Call
S3	–	Simple Storage Service
TDD	–	Test Driven Development
UI	–	User Interface
XML	–	Extensible Markup Language

Obsah

1 Úvod

V akademickém roce 2013/2014 jsem absolvoval odbornou praxi ve společnosti RAYNET s.r.o. V této společnosti již pracuji delší dobu na pozici front-end vývojáře především na platformě RAYNET CRM. Možnost absolvování bakalářské práce formou praxe pro mne byla velmi dobrou volbou, jelikož mi bylo umožněno aplikovat teoretické poznatky, kterých jsem nabyl v průběhu svých studií, v praxi.

Společnost RAYNET s.r.o.[1][2] se zabývá především vývojem Cloud CRM, ale nejedná se o jediný produkt který firma vyvíjí. Mezi přední reference této společnosti patří aplikace v oblasti zdravotnictví a financí, společnost RAYNET s.r.o. spolupracuje např. s holdingem AGEL nebo skupinou AKCENTA.

Ve společnosti působím hlavně jako front-end vývojář, avšak mé zkušenosti sahají taktéž k vývoji back-end aplikací. Na základě této flexibilnosti jsem dostal za úkol refaktoring Web end serveru RAYNET Cloud CRM, což je webový server obsluhující klientské požadavky s následným přeposláním Back end serveru k zajištěni doménové logiky.

Refaktoring Web end serveru s sebou nesl důležitá architektonická rozhodnutí, protože byla potřeba vyvinout aplikaci, která nahradí současnou a především odstraní veškeré její nedostatky. V průběhu mé praxe jsem tedy musel poznávat různé technologie, které jsou pro danou aplikaci nejvhodnější. V této práci se snažím shrnout veškerá technologická rozhodnutí, která jsem učinil a umožnit čtenáři uvědomit si veškeré výhody, které dané rozhodnutí nesly. V závěru se pokusím nastínit problematiku implementace samotného řešení. Veškerá má účast na odborné praxi se tedy sestávala z učinění technologických rozhodnutí, návrhu architektury vhodného řešení a poté samotné implementace Web end serveru.

Veškeré mé úsilí tedy bylo věnováno samotnému refaktoringu na kterém jsem strávil všechen čas na odborné praxi.

2 RAYNET Cloud CRM

RAYNET Cloud CRM[3] je hlavní produkt společnosti RAYNET s.r.o., jedná se o cloud software pro řízení vztahů se zákazníky. Software v současné době využívá množství zákazníků v České republice a na Slovensku, nicméně s jeho primárním využitím je počítáno na Americkém trhu. Z tohoto důvodu také probíhá postupný vývoj a úpravy aplikace, aby bylo možno na tomto trhu s produktem uspět. Velmi důležitým prvkem aplikace je její uživatelské rozhraní. Při analýze a návrhu řešení se obvykle v iterativním procesu vytváří UI mock-upy, které jsou navrženy do posledních detailů. Hlavní klientelou tohoto produktu jsou manažeři a obchodníci ve společnostech různých velikostí. V následujících kapitolách bych rád popsal infrastrukturu, architekturu a vývojový model RAYNET Cloud CRM.

2.1 Infrastruktura RAYNET Cloud CRM

Systém jakým je RAYNET Cloud CRM s sebou nese obrovské požadavky na výkon, stabilitu a dostupnost a tyto parametry nejsou možné splnit bez podpory kvalitní infrastruktury. Amazon web services[4] se již z počátku jevilo jako vhodné řešení pro kvalitní infrastrukturu. Platba za skutečně využité prostředky je ideální řešení postupného škálování s přibývajícím počtem uživatelů.

Servery AWS jsou situovány po celém světě a proto geografická dostupnost není problémem, obzvláště v případě RAYNET Cloud CRM, kdy je potřeba separovat instance běžící pro Evropský trh a instance běžící pro Americký trh (východní a západní pobřeží Spojených států je potřeba separovat). Pro Evropskou variantu byl vybrán server v Irsku, latence na tento server jsou dostačující i pro České, resp. Slovenské zákazníky a pohybují se v okolí 50ms. Datová centra Amazonu garantují 99.5% dostupnost. RAYNET Cloud CRM využívá dvě služby.

1. Amazon elastic compute cloud (EC2) [5]
2. Amazon simple storage service (S3) [6]

2.1.1 Amazon elastic compute cloud

Tato služba poskytuje virtualizaci serverů s možností parametrizace za pomocí webového rozhraní. V praxi změna operační paměti stroje, případně. počet jader procesoru může znamenat pár kliknutí. Služba je bezpečná a umožňuje nastavení ACL případně vlastních rozsahů IP adres pro konkrétní stroje. Platba zde probíhá za využité zdroje např. procesorový čas nebo paměť serveru. Jestliže nastane nutnost zvýšení výpočetního výkonu, postačí skrze webové rozhraní tyto parametry navýšit.

2.1.2 Amazon simple storage service

Amazon S3 poskytuje internetové úložiště pro soubory. Samotný Amazon tuto službu charakterizuje 6 vlastnostmi mezi které patří:

Bezpečnost data jsou bezpečně uložená, vlastník má plnou kontrolu nad přístupem k těmto datům

Spolehlivost garance 99.9% spolehlivosti se netýká pouze dostupnosti, ale především samotné perzistence

Škálovatelnost služba se dokáže dobře vypořádat s náhlými špičkami nejen z pohledu přenosu dat, ale např. s objemem nahraných dat

Rychlost latence pro přístup k datům je minimalizovaná a opět se dokáže přizpůsobit špičkovým potřebám

Nízká cena služba je levná a to i z obecného pohledu Cloud computingu, cena se pohybuje v desítkách centů za GB dat

Jednoduchost implementovat službu do jakéhokoliv produktu je jednoduché, Amazon připravil sadu API, které učiní tento proces přímočarý

2.1.3 Zálohování uživatelských dat

I přestože Amazon nabízí pozoruhodné možnosti pro zálohu dat, dvojí zálohování v prostředí Cloudu je považováno za nutnost. Proto je třeba uvažovat v širším kruhu a zaměřit se především na geografické oddělení zálohovaných dat z důvodu lokálních katastrof(přírodní katastrofa, teroristický útok, vojenský konflikt).

Z důvodů eliminovat datovou ztrátu jsou zálohy prováděné na geograficky oddělených místech, čímž je alespoň částečně zabráněno výše zmíněným problémům. Samozřejmostí je tato data pečlivě šifrovat a uchovávat, aby nedošlo k úniku citlivých dat. Zákazníci jsou proto chráněni systémem zálohování a šifrování pro zajištění vysoké úrovně bezpečnosti.

2.2 Vývojový model RAYNET Cloud CRM - agilní metodika Scrum

Metodika agilního vývoje Scrum[7] se využívá při vývoji RAYNET Cloud CRM, tato metodika je vhodná zejména kvůli své vlastnosti flexibilního přizpůsobeni aktuálním požadavkům na vývoj aplikace.

Vývoj proto probíhá ve 14-denním sprintovacím období jehož výstupem obvykle bývá ukázka, tzv. demo, která demonstruje veškeré úpravy případně nové vlastnosti které se v aplikaci změnily resp. byly přidány. Samozřejmostí je nasazení všech 3 klíčových rolí, tak jak je definuje Scrum.

Produktový vlastník osoba, která zodpovídá za vývoj projektu, jedná se o člověka který velmi dobře samotný produkt zná a dokáže zformulovat jeho vizi pro samotný vývoj. Na základě všech poznatků vytváří a prioritizuje produktový backlog, což je seznam požadavků, který je možné zařadit do sprintu k výrobě

Scrum master tato osoba je zodpovědná za dodržování pravidel Scrumu a obvyklé svolává porady a zajišťuje retrospektivní pohled nad samotným sprintem

Vývojový tým vývojáři, testeři, analytici a grafici zodpovědní za dodání hotových požadavků na konci sprintu

Každý den je svolaná několika minutová porada(stand-up) na které je možné objevit potenciální rizika, které ohrožují úspěšně dokončený sprint. Na těchto poradách je také možné některé požadavky ze sprintu vyjmout, protože například mohou být blokovány jiným, prioritnějším požadavkem. Na základě těchto postupů je nám umožněno flexibilně reagovat na případné změny ve vývojovém procesu.

2.2.1 Průběžná integrace a průběžná dodávka

Pro možnost využití principů agilního vývoje je velkým přínosem zavedení průběžné integrace[8]. Extrémní programováni[10] s touto technikou dokonce počítá. Tato technika funguje na principu průběžné integrace do hlavní vývojové větve i několikrát za den, čímž je minimalizovaná nutnost integrovat velké vývojové celky. Integrace velkých vývojových celků znamená zvýšené riziko zanesení chyby při integraci, která mnohdy může být těžko odhalitelná. Při každé integraci do vývojové větve je zahájen integrační proces který sestává z:

1. Kompilace aplikace
2. Kontroly kvality kódu
3. Otestování jednotkovými testy
4. Vytvoření balíčku
5. Nasazení na testovací prostředí

V případě selhání v průběhu tohoto procesu, je zbytek automaticky přerušen a všichni členové vývojového týmu jsou notifikováni. Pro průběžnou integraci využíváme nástroj Jenkins CI, tento nástroj má obrovskou škálu rozšíření které dovolují velmi nestavitelné prostředí pro průběžnou integraci.

Z důvodu využití průběžné integrace je tedy zřejmá potřeba zavést metodu programováni řízené testy(TDD)[11], které se vybízí právě ve spojení s průběžnou integraci. TDD je taky prostředek pro zdokonalení vývojového procesu agilního vývoje, jelikož umožňuje mnohem bezpečnější možnost refaktoringu, který je klíčovým prvkem agilního vývoje.

Průběžná dodávka[9] je technika kterou se pozvolna snažíme využít při vývoji RAYNET Cloud CRM. Průběžná dodávka rozšiřuje průběžnou integraci o automatizované nasazení a akceptační testy. Akceptační testy zajistí formálně definovaný stav aplikace po vývoji konkrétní nové vlastnosti. Na základě těchto testů je možno rozhodnout zdali se zadání shoduje s výstupem který dodal vývojový tým.

Nasazení aplikace je obvykle proces, který bývá komplikovaný a proto je náchylný k zanesení chyby, naneštěstí tato chyba je obvykle zavíněna lidským faktorem, z tohoto důvodu je velmi žádané automatizované nasazení, pro minimalizaci tohoto rizika.

Technika průběžné dodávky je velmi vhodná v Cloudových technologiích, protože zákazník ocení neustálý přísun nových funkcí. Průběžná dodávka ovšem vyžaduje určité prerekvizity pro samotné využití přičemž nejpodstatnější je potřeba uniformnosti testovacího a produkčního prostředí. Není možné, aby se testovací prostředí lišilo od toho reálného, protože tato situace může zapříčinit různé anomálie při testování akceptačních testu resp. nasazení samotné aplikace do produkčního prostředí.

3 Architektura RAYNET Cloud CRM

RAYNET Cloud CRM není pouze aplikace, nýbrž se jedná o komplexní systém zajišťující širokou škálu funkcionality. Tento systém neposkytuje pouze doménovou logiku, ale také např. integrační rozhraní pro externí služby nebo komplexní řešení množství komunikačních sběrnic napříč aplikaci. Na základě zřejmé robustnosti celého řešení je systém vhodně strukturován do vzájemně interreagujicích komponent a aplikací. Nosnými prvky celého systému tedy jsou:

- Cloud Manager
- CSP (Central Service Provider)
- Nginx
- RAYNET CRM Web end server
- RAYNET CRM Back end server
- RAYNET CRM klient

3.1 Cloud Manager

Jedná se o aplikaci, která má na starost chod celého systému RAYNET Cloud CRM. Je zde zajištěn kompletní systém fakturace instancí RAYNET Cloud CRM. Cloud Manager ovšem umožňuje i manuální vytváření instancí RAYNET CRM a v neposlední řadě se jedná o analytický prostředek pro monitorováni velkého množství důležitých obchodních informací. Starosti této aplikace je taktéž řízení serverů (Web end a Back end) RAYNET Cloud CRM.

3.2 Central Service Provider

Mezi základní uživatelský scénář v CRM systému patří vyhledávání. RAYNET Cloud CRM proto poskytuje vyhledávání skrze full-textový engine Apache Lucene[12]. Jelikož se jedná o potenciálně úzké místo z hlediska využití systémových zdrojů, existuje aplikace která má na starost zejména:

1. Veškeré integrační služby třetích stran, např.: načítání informací o firmě z obchodního rejstříků nebo orientační kreditabilita firmy
2. Fulltextové vyhledávání
3. Odesílání e-mailů pro uživatelé RAYNET Cloud CRM

CSP existuje především z důvodu odstranění těsných vazeb mezi externími systémy a RAYNET Cloud CRM.

3.3 Nginx

Nginx[13][14] v RAYNET Cloud CRM přebírá roli výkonné a důležité reverzní proxy. Z výkonového hlediska se jedná o perfektně škálovatelné řešení, které vyhovuje potřebám neustále se rozšiřující klientele cloudové aplikace. Nginx v roli reverzní proxy umožňuje snadné směrování URL na instance samotné aplikace RAYNET CRM a webových stránek společnosti. Webové stránky a instance RAYNET CRM(instance RAYNET CRM mají unikátní prefix v URL adrese dle názvu samotné instance) běží na společné doméně a platí zde velmi jednoduchá pravidla pro směrování URL na konkrétní aplikaci.

3.4 RAYNET CRM Web end server

RAYNET Cloud CRM je webová aplikace pro jejíž chod je nezbytný webový server. Předmětem mé praxe ve společnosti RAYNET s.r.o. byl právě refaktoring této části aplikace. Tato část má na starost obsloužit požadavky uživatelů skrze HTTP protokol a poslat je dál Back end serveru. Jedná se tedy o velmi jednoduchou formu proxy a okrajově taktéž systému pro vyvažování zátěže. Web end server umožňuje formovat klientské požadavky pro další zpracování.

Před samotným refaktoringem byl Web end server samostatná aplikace která v sobě zahrnovala HTTP server Apache Tomcat[15]. Tento webový server je otevřenou implementací Java Servlet API[16]. Ve spojení s Apache Tomcat byl využit aplikační rámec Spring MVC[17] umožňující především správu sezení, směrování URL a vkládání závislostí[18].

Obrovskou nevýhodou předchozího řešení byla nutnost vytvářet s každou instancí RAYNET Cloud CRM nový aplikační kontext, tento aplikační kontext byl minimalizován na nejmenší možnou velikost, ale i přes tento fakt si žádal část systémových prostředků, které se ve velkém měřítku(desítky tisíc instancí RAYNET Cloud CRM) podílí na nereálném využití především operační paměti.

Předchozí implementace tedy byla chápána jako technický dluh, který ve velké míře mohl přispět k neúnosnosti celého systému při enormním množství uživatelů, na které je při přechodu na mezinárodní trh potřeba myslet. Čas potřebný pro start tohoto serveru byl v řádech sekund na instanci, při vydání nové verze RAYNET Cloud CRM se jednalo o úzké místo z hlediska potřebného času, tato restrikce se navíc úměrně navyšuje s počtem vytvořených instancí.

3.5 RAYNET CRM Back end server

Back end server je klíčovou komponentu celého systému poskytující doménovou logiku. Těchto serverů může být větší množství, čímž je zajištěno spolehlivé vyřízení velkého množství paralelních požadavků. Aplikace je postavena na aplikačním rámci Spring Framework[17] pro snadnou konstrukci komplexních objektů a jejich závislosti za pomocí návrhového vzoru vkládání závislostí. Back end server na přímo komunikuje se všemi datovými úložištěmi RAYNET Cloud CRM.

Perzistence dat není zajištěná pouze standardní SQL databázi, nýbrž je použito vícero technologií, za zmínku stojí především:

PostgreSQL [19] datové úložiště pro entitní objekty

MongoDB [20] nerelační databáze pro některé data, které nejsou součásti entitních objektů, např. integrační data zaslaná z Cloud Manageru

MongoDB GridFS úložiště pro dynamické soubory např. loga, fotografie atd.

3.6 RAYNET CRM klient

Cloud služba RAYNET CRM je založená na neustálé možnosti připojení k aplikaci z internetu bez nutnosti instalace aplikace na zařízení. Na základě tohoto faktu je nutno zajistit pohodlný přístup za pomocí uživatelsky přívětivého klienta implementovaného v internetovém prohlížeči. Tato aplikace je formou tzv. chytrého klienta a tedy veškeré služby doménové logiky poskytuje server s kterým klient komunikuje skrze http protokol za pomocí principů REST[21].

Klient RAYNET Cloud CRM klade důraz na uživatelskou přívětivost a jednoduchost a proto je naprosto esenciální aby se jednalo o aplikaci na jedné webové stránce. Pro komunikaci a navigaci v aplikaci je použitá technologie AJAX[22] umožňující asynchronní načítání dat ze serveru bez nutnosti pře-načtení celé stránky. Takto komplikovaný systém v sobě nese nevýhodu v podobě složité implementace a udržovatelnosti. Klasický přístup vývoje webových aplikací je proto pro takový typ aplikace značně nevhodný, jelikož neposkytuje potřebnou robustnost která zajistí přehlednost, spolehlivost a rozšiřitelnost aplikace.

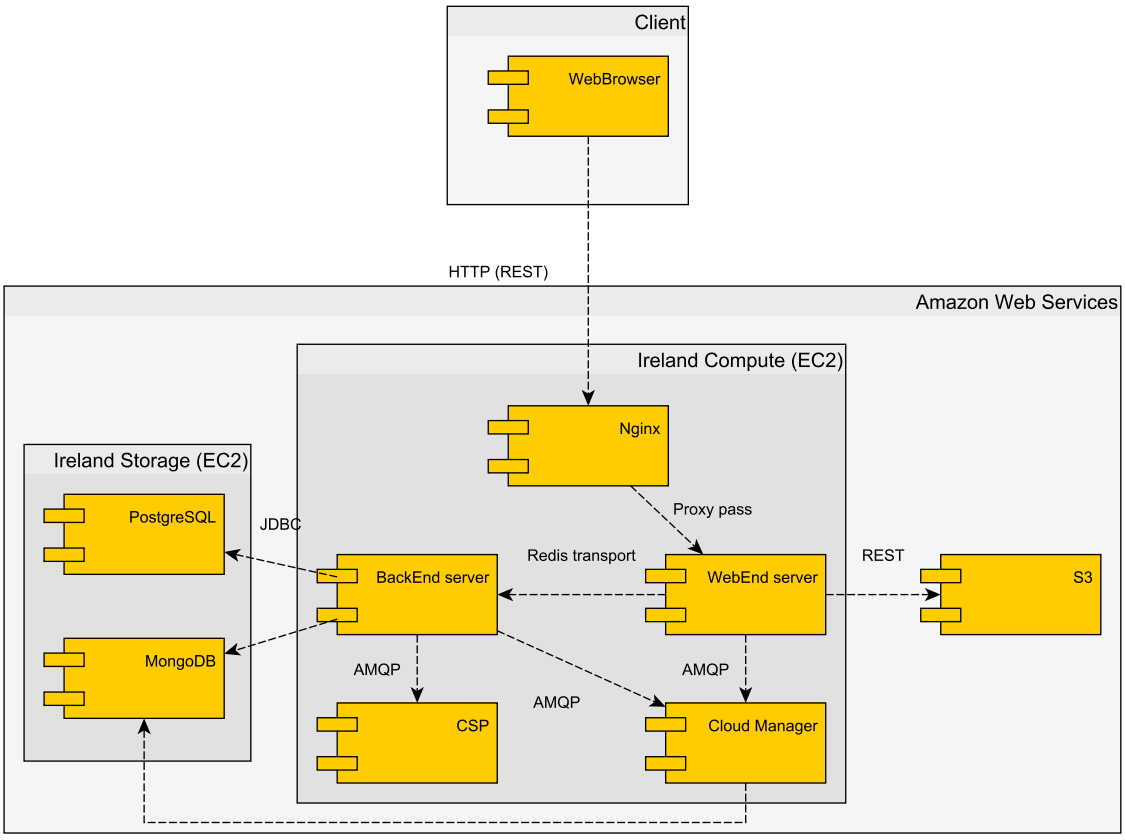
Pro složité GUI aplikace se mnohem více hodí vývoj na základě komponent, který ovšem není nativně podporován při vývoji webových aplikací. RAYNET Cloud CRM proto využívá aplikačního rámce ExtJS[23], který v základní podobě nabízí množství předpřipravených GUI komponent mezi které patří např. data view, panel, button. Tyto komponenty jsou snáze rozšiřitelné a stylovatelné, aby byl zajištěn unikátní vzhled každé aplikace postavené na bázi tohoto aplikačního rámce. Aplikační rámce ExtJS není ovšem pouze sada předpřipravených komponent, nýbrž poskytuje robustní základ pro architekturu komplikovaných webových aplikací. Samozřejmostí tohoto aplikačního rámce je zajištění základních možností, které jsou očekávané jako např. komunikace se serverem, logování, práce s formátem JSON a mnoho dalších důležitých funkcí a vlastností. Aplikační rámec ExtJS závisí na použití kaskádových stylů pro definování vzhledu aplikace. Z tohoto důvodu využíváme technologii Sass[24], která umožní kompilaci kaskádových stylů, před samotným použitím. Mezi hlavní přínosy této technologie patří:

Hierarchické selektory selektory pro kaskádové styly je možné jednoduše zanořovat do sebe, tato vlastnost zkracuje a zpřehledňuje výsledný kód jelikož dlouhé selektory není potřeba opisovat vícekrát

Proměnné Mnohdy je potřeba vícenásobného použití např. barvy písma v kaskádovém stylu, využití proměnných tyto scénáře ulehčuje a navíc umožňuje parametrizaci vzhledu (např. barevná schémata)

Mix-in mixování částí kódu dohromady, tato funkce umožní např. stejné části kaskádových stylů smíchat dohromady a získat naprosto nový vzhled pro novou komponentu

Využití čistého JavaScriptu na straně klienta sebou nese i určité nevýhody. Mezi největší nevýhodu současné technologie jednoznačně patří obtížné psaní jednotkových testů a volnost samotného jazyka. JavaScript je dynamický jazyk založený na prototypové implementaci objektově orientovaného programováni na základě čeho umožňuje snadné zanesení špatných návyku ze strany programátora. Mohou tedy vznikat těžko odhalitelné chyby a refaktoring se stává nebezpečným. V budoucnu se jako vhodná alternativa jeví programovací jazyk Dart s využitím specifikace Web Componenets[25].



Obrázek 1: Diagram nasazení pro základní komponenty systému

4 Implementace nového Web end serveru

S vidinou expanze RAYNET Cloud CRM na mezinárodní trh bylo nutno analyzovat úzká místa současného stavu aplikace se snahou tyto problémy minimalizovat, refaktoring klíčových prvků nebyla jediná potřeba pro plynulý přechod na mezinárodní trh. Doménová logika je napříč veškerých obchodních společností po světě taktéž velmi specifická a proto bylo potřeba vzít v úvahu regionálně specifické požadavky, které vyplynuly z přechodu do mezinárodního prostředí.

Klíčový aspekt ovšem nadále zůstává spolehlivost a responzivnost aplikace napříč různými kontinenty pro obrovské množství potenciálních zákazníků. Tento požadavek s sebou jednoznačně nese nutnost tlačit požadavky na systémové zdroje na minimum. Stav současného Web end serveru je obrovským rizikem pro přechod na mezinárodní trh a to především z důvodu neakceptovatelných nároků na systémové prostředky(tj. především operační paměť), které obvykle lineárně narůstají s počtem vytvořených instancí RAYNET Cloud CRM. Jelikož je RAYNET Cloud CRM řešení cloudové, požadavek minimalizovat čas výpadku aplikace je na základě tohoto faktu zřejmá, tento fakt ovšem přímo ovlivňuje schopnost průběžné dodávky, které je pro uživatelský zážitek důležitá. Z tohoto důvodu je nutné snížit čas startu nového Web end serveru na minimum.

Vedlejším efektem refaktoringu Web end serveru bude odstranění nežádoucích limitů Servlet API a podstatné zjednodušeni vrstvy, která slouží v podstatě pouze jako proxy server. Implementace nového Web end serveru musí jednoznačně být přímočará a snadno integrovatelná pro nové vývojáře.

4.1 Komunikace mezi Web end a Back end serverem

Je zapotřebí ustanovit komunikační vrstvu mezi Web end a Back end serverem, tato vrstva je kritická pro celý systém, je tedy nutné zajistit základní požadavky:

- rychlost
- flexibilnost
- škálovatelnost
- nízké datové zatížení

Současná architektura komunikační vrstvy se jeví jako spolehlivá a splňuje veškeré výše uvedené požadavky. Na základě statistik dlouhodobého používání těchto technologií neexistuje pádný důvodu pro potřebu renovace. Z důvodu již poměrně rozsáhlého API, které Back end server poskytuje by bylo vhodné zvážit zdali je nutno refaktoring v této oblasti vůbec podstoupit a zdali by tato vynaložená snaha nezpůsobila zvýšenou šanci k chybovosti aplikace. Musíme rozhodně uvažovat z hlediska rozsahu úprav, který přímo úměrně ovlivní cenu nového řešení.

Aktuální kandidáti pro implementaci komunikační vrstvy mezi Web end a Back end serverem tedy zůstává kombinace technologií:

Redis [26] paměťová databáze pro předávání zpráv zajišťující potřebnou rychlost a škálovatelnost

Google Protocol Buffers [27] umožňující uniformní zprávy napříč platformami zajišťující potřebnou flexibilnost

Snappy [28] nástroj pro kompresi zpráv s přijatelným kompresním poměrem neohrožující rychlost výsledného řešení

4.1.1 RabbitMQ

RabbitMQ[29] je systém pro příjímání a publikaci zpráv napříč různými systémy, tato technologie není přímo obsažena v komunikační vrstvě mezi Web end a Back end serverem, ale je vhodné ji zmínit jako dobrého kandidáta pro komunikaci.

RabbitMQ je v RAYNET Cloud CRM hojně využívaná technologie integrující především velké množství aplikací, které jsou součástí celého systému. Je tedy více než vhodné zauvažovat zdali by úsilí vynaložené k použití RabbitMQ jako integračního rozhraní mezi Web End a Back End serverem nestálo za možnosti, které tento systém nabízí a to především z důvodu pozitivních zkušeností v nasazení jako integračního rozhraní mezi autonomními systémy.

RabbitMQ implementuje protokol AMQP[30] (Advanced Message Queuing Protocol) jenž představuje jakýsi standard pro Message Broker[31] systémy jakým je i RabbitMQ. Tento protokol definuje pojmy jako

Queue jedná se o frontu, která slouží pro příjem zpráv

Exchange jednotné rozhraní pro odeslání zprávy

Routing definuje pravidla (cesty) mezi exchanges a queues, resp. snaží se vytvořit správnou cestu pro zprávu odeslanou skrze exchange a poté přijatou na konkrétní frontě (front pro příjem jedné zprávy může být více)

Zprávy je možné potvrzovat pomocí ACK resp. NACK. Implementace RPC (remote procedure call) je velmi triviální a omezuje se pouze nutností vygenerovat klientský identifikátor na základě kterého se bude odesílat odpověď na předem vytvořenou frontu. V praxi je tedy potřeba ze strany klienta zaslat zprávu jejíž hlavička bude obsahovat replyTo atribut a vytvořit příslušnou frontu ze které je potřeba poslouchat odpověď. Po příjmu zprávy serverem se odpověď zašle na předem vytvořenou frontu, jejíž identifikátor je obsažen v hlavičce zprávy.

Jelikož RabbitMQ umožňuje pokročilou formu směrování zpráv, je velmi snadné implementovat spoustu pokročilých funkcí jako např. monitoring nebo hromadnou integraci a proto se tato technologie hodí jako integrační sběrnice. Velkou výhodu v tomto ohledu je zajisté jeho implementace napříč různými programovacími jazyky a platformami(JAVA, .NET, C++, Python).

Koncept implementace RPC za pomocí RabbitMQ může jednoduše řešit výpadky a to nejen aplikační nebo síťové, ale především výpadky spojené s vydáváním nové

verze Back end serveru. Integrační sběrnice zůstává pořád funkční a to i v případě kdy nastane výpadek Back end serveru, veškeré zprávy se hromadí a po opětovném navázání spojení Back end serveru s RabbitMQ může systém velmi pružně reagovat na odeslané požadavky v případě kdy je to potřeba tj. nenastal timeout požadavku. Jestliže se povede minimalizovat dobu restartu Back end serveru na minimum(v řádech sekund) může to znamenat snadné nasazení nové verze Back end serveru bez nutnosti omezení zákazníka, ten samotný výpadek nemusí ani zaznamenat, případně jeho požadavky mohou být zpracovány s drobným zpožděním.

RabbitMQ ovšem s sebou nese poměrně velkou režijní činnost spojenou především s vytvářením spojení a jeho udržením, jeho nasazení tedy není úplně vhodné u skriptovacích jazyků a to především v případě kdy je rychlost celého systému klíčový faktor. Přestože se RabbitMQ jeví jako vhodný kandidát pro komunikační vrstvu mezi Web end a Back end serverem, jeho nasazení bylo zváženo a zamítnuto a to především z důvodu režie spojené s vytvořením a údržbou spojení a celkový koncept není dostatečně jednoduchý a přímočarý pro výkonově kritickou část aplikace jakou komunikační vrstva bezesporu je.

4.1.2 Redis

Redis je jednoduchá paměťová databáze pro uchovávání dat ve formátu klíč — hodnota. Její současná implementace umožňuje i perzistenci na disku avšak hlavní výhoda této technologie tkví v její rychlosti a jednoduchosti, která je zajištěna právě za pomocí perzistence v paměti. Tato vlastnost redis staví mezi přední technologie v oblastí integrací. Databáze umožňuje i jednoduchou formu replikace(Master - Slave), tato vlastnost ale při implementaci nebyla využita.

Redis dokáže k datům přistupovat v různých typech např. fronty, hash mapy a seznamy. Pro integraci jsou ovšem důležité dva typy

- Fronta s využitím left push a blocking right pop
- Kanál s využitím publish / subscribe

Pro svou jednoduchost byl pro komunikaci mezi Web end a Back end serverem RAYNET Cloud CRM vybrán právě Redis s využitím komunikace za pomocí front. Režie spjatá s vytvořením spojení a jeho udržením je malá a proto se jeví i jako ideální kandidát jako komunikační sběrnice i při využití skriptovacích jazyků. Implementace je velmi snadná pro různé programovací jazyky a platformy, výrobce ovšem doporučuje pro produkční prostředí Redis provozovat na Linuxovém serveru. Za zmínku také stojí využití této technologie velkými aplikacemi jakou je např. Pinterest.

4.1.2.1 Komunikační rozhraní s využitím front

Volání API s využitím front funguje na principu zaslání zprávy do seznamu zleva. Tato zpráva musí obsahovat atribut s identifikací klíče za pomocí kterého server zašle odpověď. Veškerá data vložená do Redisu jsou obyčejné řetězce a proto je potřeba využít vrstvu obalující data uložené v databázi(v

případě RAYNET Cloud CRM se jedná o Protocol Buffers) a tyto data v sobě budou držet informaci o identifikátoru klíče pro odpověď. Klient po zaslání zprávy využije blokující volání na výběr klíče zprava za pomocí identifikátoru klíče který zaslal pro odpověď. Toto volání je blokující po určitou dobu(v řádu sekund), při vypršeni časového limitu klient situaci musí vyhodnotit jako nevrácenou odpověď od serveru. Server zprávu zpracuje a vrací odpověď do seznamu zleva, tento seznam je identifikován v těle zprávy.

Tento postup opět zajišťuje odolnost vůči výpadkům v komunikaci, které mohou být způsobeny např. výpadkem serveru z důvodu vydání nové verze Back end server. Technologie tedy velmi jednoduše umožňuje rozpojení klíčových komponent systému nezávisle na potřebě.

4.1.2.2 Komunikační rozhraní s využitím kanálů Tento postup je velmi podobný jako komunikace za použití front, jediný rozdíl je namísto v odeslání a příjmu zprávy na jednom resp. dvou seznamech, je zpráva zaslaná do kanálu na který může poslouchat více serverů. Tato technika velmi jednoduše umožňuje implementovat jednoduchý load balancing. Back end serverů může běžet najednou více, které budou společně přijímat požadavky od stejného klienta a na základě vytíženosti konkrétního serveru, se mohou ostatní požadavky směřovat na jiný server, který je v konkrétním okamžiku vytížen méně.

4.1.3 Google Protocol Buffers

Jedná se o mechanizmus serializace strukturovaných dat. Mechanizmus byl vyvinut společností Google a je používán v širokém spektru Google aplikací. Serializované data jsou binární a proto je tato technologie velmi vhodná jako alternativa k datově velmi neefektivnímu formátu XML. Mezi přední vlastnosti Google Protocol Buffers patří:

- Binární serializace
- Zpětná kompatibilita zpráv
- Platformová nezávislost

Veškeré typy zpráv které aplikace používá je potřeba popsat v jednoduchém textovém formátu a poté se provede kompilace těchto zpráv pro využití v konkrétním programovacím jazyku. Generované třídy jsou zodpovědné za korektní serializaci a deserializaci přenášených dat. V nejjednodušší formě se tedy na jedné straně data zabalí do klasické třídy, která odpovídá formátu specifikované zprávy. Na straně druhé se provádí deserializace, která binární data převádí zpět na objekty. Tato technologie vybízí srovnání s ostatními hojně využívanými formáty dat jako je XML nebo JSON. Oproti XML je ovšem Google Protocol Buffers výrazně jednodušší, avšak stále se jedná o velmi bezpečnou a striktní datovou specifikaci jakou např. JSON nemůže poskytnout. Zřejmý je taktéž nízký datový objem a to již ze samotného principu přenosu binárních dat.

Komunikace mezi Web End a Back End serverem RAYNET CRM klade obrovské požadavky na rychlost, spolehlivost a nízké datové zatížení. Zde se technologie Google Protocol Buffers jeví jako velmi vhodný kandidát a to z hned z několika důvodů:

Rychlost Google Protocol Buffers poskytují velmi malou režii spojenou s serializací a deserializací dat, čímž splňují hlavní podmínku.

Nízké datové zatížení Serializace je binární a zprávy v sobě nesou jen velmi malé množství meta informací potřebných pro komunikaci, technologie proto klade malé požadavky na velikost samotné zprávy. Velikost je taktéž minimalizovaná i při zasílání velkého množství malých zpráv z důvodu jednoduchosti protokolu.

Multiplatformost Není vyloučená budoucí potřeba přechodu na jinou platformu, tento fakt ovšem nijak nelimituje využití technologie Google Protocol Buffers jelikož je multiplatformní.

Typová bezpečnost Veškeré zprávy jsou staticky typované, tento fakt přispívá k bezpečnosti využití technologie z hlediska uniformního API mezi Web End a Back End serverem. Specifikace je jasně daná v definici zprávy.

4.1.4 Snappy

Je zřejmé, že komunikační rozhraní mezi Back End a Web End serverem, je citlivé na velikost přenášených dat, přeci jen se jedná o hlavní komunikační kanál mezi těmito komponentami, které jsou základním stavebním prvkem RAYNET Cloud CRM.

Komprese dat, které mezi těmito servery putují je tedy nutná. Je třeba ovšem brát ohled na rychlost samotného řešení. Komprese a následná dekomprese dat je vždy výpočetní operace, která si klade nároky na systémové zdroje, čímž je nutné zajistit, aby tyto požadavky na výpočetní výkon minimalizované.

Na základě těchto informací je v RAYNET Cloud CRM využita knihovna Snappy vyvinuta společnosti Google. Samotní autoři si nekladou za cíl maximálního kompresního poměru nýbrž dosažení velmi vysoké rychlosti komprimace s zachováním dostačujícího kompresního poměru. Tato hodnota může být až několikanásobně nižší než např. při využití komprese gzip. Výchozí implementace této knihovny je v jazyce C++, v RAYNET CRM je ovšem využita alternativa na platformě JAVA.

4.2 Architektura nového Web End serveru

Web end server by měl plnit následující úlohy:

1. Vstupní bod pro HTTP požadavky ze strany klienta
2. Mapování požadavků na volání API
3. Zpracování a obsluha HTTP sezení pro přihlášení uživatele

Při bližší úvaze dojdeme k závěru, že se jedná o reverzní proxy server umožňující přihlášení uživatelů s možnosti jednoduchého vyvažování zátěže. Předchozí architektura ve formě těžkotonážního HTTP JAVA serveru s vlastním aplikačním kontextem pro každou instanci je tedy zbytečně komplikované řešení, které spotřebovává značné

množství systémových prostředků. Nové řešení by tedy mělo být především jednoduché a spolehlivé. Hlavní požadavky na novou architekturu jsou nízké nároky na paměť, které ideálně nebudou narůstat se zvětšujícím se počtem vytvořených instancí. Dalším důležitým faktorem je rychlost náběhu Web End serveru a to z důvodu potřeby průběžné dodávky a častého vydávání nových verzí aplikace. V neposlední řadě je potřeba zajistit škálovatelnost aplikace při velkém vytížení, které nemusí být pouze špičkového charakteru.

4.2.1 Nginx v roli proxy ve spojení s php-fpm

Nginx jsem již v této práci popisoval, jedná se o výkonný proxy server, který může sloužit i jakožto webový server. Ve společnosti RAYNET s.r.o. s tímto serverem máme bohaté a pozitivní zkušenosti a to především z důvodu jeho obrovské stability a výkonu. Na základě těchto zkušeností(především z nasazení na webových stránkách společnosti) nelze pochybovat o jeho schopnosti nahradit webový server pro samotnou aplikaci. V úvodu kapitoly o architektuře jsem zmínil požadavky na škálovatelnost a paměťovou nenáročnost a v obou těchto aspektech nginx naprosto exceluje. Je potřeba připomenout taktéž fakt, že nginx se hodí jako primitivní forma vyvažovače zátěže, což může mít velký vliv na propustnost požadavků. Nginx samotný ovšem není dostatečný prostředek pro vybudovaní stabilního a výkonného Web end serveru, který bude sloužit pro mapování požadavků API pro Back end server - je potřeba využít skriptovacího jazyka pro jednoduchou logiku celého systému a zajištění komunikace s Back end serverem. Jakožto skriptovací jazyk jsem zvolil PHP a to zejména z důvodů jednoduchosti, která ovšem neubírá na spolehlivosti v případě využití moderních aplikačních rámců. Zapojení nových vývojářů v případě jazyka PHP není až tak velký problém.

Moderní vývoj jednoduchý webových aplikací v PHP je velmi efektivní jelikož existuje řada velmi spolehlivých aplikačních rámců, které vývoj zabezpečuji a zjednodušuji za zmínku stojí např. Zend Framework nebo Symfony, taktéž se z podstaty samotného jazyka vývoj urychluje a s využitím jednotkových testů to nemusí být nutně na úkor bezpečnosti či chybovosti kódu. Nginx samotný ovšem spouštění PHP skriptů nepodporuje a proto je za potřeby využit technologie CGI. Php-fpm[32] je FastCGI manažer procesů schopný obrovské propustnosti z hlediska zpracování požadavků. Konfigurace php-fpm spočívá hlavně v nastavení velikosti poolu vláken, které budou využity při vyřizování požadavků.

Veškerá podstata využití nginx jako webového serveru s php-fpm pro spouštění skriptů na straně serveru tedy spočívá v škálovatelnosti, paměťové nenáročnosti a jednoduchosti začlenění nových vývojářů do vývojového procesu. V předchozích kapitolách jsem taktéž zmínil nutnost rychlého startu samotného serveru. Tento čas je naprosto minimalizován a to především pouhou potřebou restartu nginx. Je nutné si taktéž uvědomit, že je možné velmi urychlit samotnou exekuci PHP skriptu nejrůznějšími nástroji jako je např. xcache. Kompilace skriptu tedy nemusí probíhat při každém požadavku, nýbrž se bude jednat o pouhou exekuci již předkompilovaného skriptu. Řešení umožňuje bezstavový přístup, výpočetní režie spojená s nutnosti udržovaní stavu se minimalizuje, toto je zřejmá výhoda oproti starému řešení Web End serveru, kdy byl vytvořen pro každou

instanci jeden malý aplikační kontext. Paměťová náročnost tedy není lineárně úměrná s narůstajícím počtem instancí.

Jelikož jsou skripty spouštěné s každým požadavkem znovu a pouze některé prostředky zůstávají v paměti jako např. připojení k databázi redis atp., je možná alespoň částečná obrana vůči paměťovým únikům. Web End server by ovšem ve své podstatě měl být velmi jednoduchý a samotný kód neměl umožňovat nebezpečné prvky ze strany programátora. Hlavní rozšířitelnost Web end serveru spočívá v definování nových volání API na Back end server, které narůstají s novými vlastnostmi, které jsou poskytnuty klientovi.

4.2.2 Aplikační rámec Slim

Z analýzy požadavků vyplynula potřeba využití PHP jakožto skriptovacího jazyka na straně serveru pro nový Web End server. Samotný jazyk PHP je velmi benevolentní vůči vývojáři a umožňuje aplikaci nejrůznějších zlých zlozvyků ze strany vývojáře. RAYNET CRM Web End server musí být robustní komponenta celé aplikace postavena na pevném základu. Tímto základem byl zvolen aplikační rámec Slim. Aplikační rámec Slim[33] je zajímavá technologie, která cílí využití nejnovějších vlastností jazyka PHP. Tyto vlastnosti napomáhají tomuto aplikačnímu rámci dosáhnout efektivity při vývoji webové aplikace se společným zachováním jednoduchosti celé technologie. Aplikační rámec Slim těží z moderních vlastnostní[34][35] jayzka PHP.

4.2.2.1 Jmenné prostory PHP před verzí 5.3 trpělo absencí jmenných prostorů, tento fakt nutil vývojáře vytvářet názvy tříd pomocí tzv. podtržítkové konvence, tj. zanořené třídy v adresářích jsou v celém názvu odděleny za pomocí podtržítek. Výsledkem byl velmi dlouhý název třídy, které byly nesnadno vyhledatelné pomocí vyhledávacích funkcí tříd vestavěných v IDE

4.2.2.2 Lambda funkce Zavedení anonymních funkcí a closures v jazyce napomáhá implementaci řízenou událostmi. V některých případech se tato technologie hojně využívá např. v aplikačním rámci Slim.

4.2.2.3 PSR0 autoloading tříd Standard PSR0 napomáhá sjednocení adresářové a souborové struktury tříd napříč projekty, tímto standardem se řidí např. Composer.

Instalace závislostí do PHP projektu zajišťuje manažer závislosti Composer. Manažer je zodpovědný za stažení veškerých závislostí do projektu z webového repozitáře. Composer je obdobou systému jako je např. Gradle nebo Maven. Samozřejmostí je nutnost zavedení jednotné adresářové struktury napříč projektem.

4.2.3 Využití aplikačního rámce Slim

Aplikační rámec Slim je minimalistický a vývojáři umožňuje velmi volnou ruku v návrhu samotné aplikace, tento aplikační rámec řeší základní problémy kterým je nucen vývojář ve většině případů čelit. Bylo proto nutné položit pro aplikaci pevný základ.

Aby aplikace zůstala jednoduchá, ale přesto snadno rozšiřitelná rozhodl jsem se implementovat návrhový vzor *Front Controller*[36]. Tento návrhový vzor vytváří jednotný přístupový bod pro webovou aplikaci a deleguje daný požadavek na tzv. aplikační controller, který už má na starost samotnou vykonávanou akci pro daný požadavek. V aplikačním rámci Slim tohoto může být využito ve spojení s třídou *Router*, která umožňuje snadné mapování HTTP požadavku(GET, POST, PUT, DELETE) na konkrétní lambda funkci. Front controller je tedy zodpovědný za načtení veškerých aplikačních controlleru a delegaci konkrétních akcí na základě směrovacích pravidel.

Aplikační rámec Slim obsahuje techniku zvanou *Middleware*, kdy každý požadavek může být modifikován zřetězeným voláním jednotlivých Middleware a každý jeden z nich předává kontrolu dalšímu. Následující volaný Middleware obsahuje celý kontext aplikačního rámce(tedy i HTTP požadavku). V případě kdy Middleware nezavolá metodu *call*, čímž předává kontext aplikace další vrstvě, dochází k přerušení volání další vrstvy a požadavek je ukončen. Middleware lze taktéž svázat s konkrétní cestou nebo skupinou cest. Této techniky lze velmi dobře využít např. u bezstavových požadavků, které se autentizují za pomocí HTTP basic authentication. Pro obsluhu čistě AJAX volání např. z mobilního klienta, kde je nutnost zachovat bezstavovost komunikace je možno využit právě HTTP basic authentication kdy pro danou skupinu URL adres existuje *AuthenticationMiddleware*, jehož starostí je uložit z požadavku informace o přihlášeném uživateli do třídy implementující rozhraní *IAuthenticationProvider*. Toto rozhraní v aplikaci slouží pro jednotný přístup k přístupovým údajům o uživateli. Může nastat situace, kdy se přístupové údaje pro komunikaci s Back end serverem dotazují z HTTP session, ale právě za pomocí tohoto rozhraní je konkrétní implementace pro přihlašovací údaje skrytá.

Příklad pro řešení dotazování přihlašovacích údajů taktéž velmi dobře demonstruje využití návrhového vzoru *Inversion of control*[18]. Každý volaný požadavek za pomoci Middleware určí jaká autentizace se pro požadavek používá a na základě tohoto rozhodnutí uloží v kontextu aplikačního rámce konkrétní třídu implementující rozhraní IAuthenticationProvider. V aplikaci tedy existují různé implementace tohoto rozhraní jako je např. *SessionAuthenticationProvider*, která poskytuje autentizační údaje z HTTP session, nebo *BaseAuthAuthenticationProvider* která poskytuje autentizační údaje za pomocí HTTP basic authentication.

5 Závěr

5.1 Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné v průběhu odborné praxe

Již v úvodu samotné práce jsem zmínil informaci, že odborná praxe pro mne je jistou formou aplikace mých teoretických poznatků, které jsem nabyl v průběhu studia v praxi. Teoretický základ poskytnutý v průběhu studia byl pro řešení mého problému pouhý základ, ke kterému bylo zapotřebí přidat obrovské množství praktických zkušeností, abych byl schopen zadaný úkol realizovat. Naneštěstí předchozí praktické zkušenosti, které nabízela škola byly omezené a právě proto tato absence mohla zapříčinit můj neúspěch na zadaném úkolu. Naopak teoretický základ byl pevný a umožnil mi snadnou orientaci při studii technologií, které jsem pro vypracováni úkolu potřeboval.

5.2 Znalosti či dovednosti scházející v průběhu odborné praxe

Většina technologií, které jsem při praxi využíval pro mne byla nová a při studiu jsem se s nimi nesetkal. Bylo tedy zapotřebí velké množství času investovat právě do samovzdělání v těchto technologiích. Jedná se především o tyto technologie:

- RabbitMQ
- Redis
- Google Protocol Buffers
- Nginx + php-fpm

Velmi dobré teoretické znalosti informatiky a softwarového inženýrství, které jsem nabyl při mém studiu, mi ovšem dovolily relativně plynule a rychle tyto technologie prostudovat.

5.3 Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Refaktoring Web end serveru pro RAYNET Cloud CRM byl dokončen a úspěšně nasazen v testovacím prostředí. Současný stav umožňuje komunikaci pro mobilního klienta RAYNET Cloud CRM, veškeré cíle které jsem kladl na začátku pro novou aplikaci byly splněny. Reálný přínos ovšem bude měřitelný až s průběhem času, kdy dojde k nasazení RAYNET Cloud CRM na americký trh.

Věřím, že jsem položil pevné základy pro toto kritické místo v systému a přínosy budou zřejmé nejen z technologického hlediska, ale také např. z finančního, kdy tento refaktoring umožní nikoliv pouze zlevnit současný provoz, nýbrž připravit systém na enormní zátěž a poskytnout tedy službu obrovskému množství uživatelů po celém světě.

Odborná praxe mi umožnila enormní posun v mé dosavadní profesní kariéře a věřím, že Vysoká škola báňská byla vhodným zprostředkovatelem.

6 Reference

[1] RAYNET S.R.O. - *O společnosti RAYNET* [online]. [citováno 20-04-2014] Dostupné z: https://raynet.cz/o-spolecnosti-raynet.html

[2] RAYNET S.R.O. - *Reference vybraných klientů* [online]. [citováno 20-04-2014] Dostupné z: https://raynet.cz/reference.html

[3] RAYNET S.R.O. - *Co je RAYNET CRM* [online]. [citováno 20-04-2014] Dostupné z: https://raynet.cz/co-je-raynet-crm.html

[4] AMAZON MEDIA ROOM - *Amazon Web Services Launches* [online]. [citováno 20-04-2014] Dostupné z: http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=830816&highlight=

[5] AWS - *Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting* [online]. [citováno 20-04-2014] Dostupné z: http://aws.amazon.com/ec2/?nc1=h_l2_cn

[6] AWS - *Amazon Simple Storage Service (S3) - Online Cloud Storage for Data & Files* [online]. [citováno 20-04-2014] Dostupné z: http://aws.amazon.com/s3/

[7] *Core Scrum – Values and roles* [online]. [citováno 20-04-2014] Dostupné z: http://www.scrumalliance.org/why-scrum/core-scrum-values-roles

[8] FOWLER, Martin *Continuous Integration* [online]. [citováno 20-04-2014] Dostupné z: http://martinfowler.com/articles/continuousIntegration.html

[9] FOWLER, Martin *Continuous Delivery* [online]. [citováno 20-04-2014] Dostupné z: http://martinfowler.com/bliki/ContinuousDelivery.html

[10] *What is Extreme Programming?* [online]. [citováno 20-04-2014] Dostupné z: http://xprogramming.com/what-is-extreme-programming/

[11] JANZEN, David S.; SAIEDIAN, Hossein. *Test-driven development: Concepts, taxonomy, and future direction.* [citováno 20-04-2014] Dostupné z: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1034&context=csse_fac&sei-redir=1

[12] THE APACHE SOFTWARE FOUNDATION *Apache Lucene* [online]. [citováno 20-04-2014] Dostupné z: https://lucene.apache.org/

[13] NGINX INC. *The High Performance Reverse Proxy, Load Balancer, Edge Cache, Origin Server* [online]. [citováno 20-04-2014] Dostupné z: http://nginx.com/

[14] PCMAG DIGITAL GROUP *reverse proxy Definition* [online]. [citováno 20-04-2014] Dostupné z: http://www.pcmag.com/encyclopedia/term/50498/reverse-proxy

[15] THE APACHE SOFTWARE FOUNDATION *Apache Tomcat* [online]. [citováno 20-04-2014] Dostupné z: http://tomcat.apache.org/

[16] ORACLE *Java Servlet Technology Overview* [online]. [citováno 20-04-2014] Dostupné z: http://www.oracle.com/technetwork/java/overview-137084.html

[17] JOHNSON, Rod, Juergen HOELLER, Alef ARENDSEN, Colin SAMPALEANU, Rob HARROP, Thomas RISBERG, Darren DAVISON, Dmitriy KOPYLENKO, Mark POLLACK, Thierry TEMPLIER, Erwin VERVAET, Portia TUNG, Ben HALE, Adrian COLYER, John LEWIS, Costin LEAU, Mark FISHER, Sam BRANNEN, Ramnivas LADDAD. *The Spring Framework: Reference Documentation* [online]. [citováno 20-04-2014] Dostupné z: http://docs.spring.io/spring/docs/2.5.3/reference/index.html

[18] FOWLER, Martin *InversionOfControl* [online]. [citováno 20-04-2014] Dostupné z: http://martinfowler.com/bliki/InversionOfControl.html

[19] THE POSTGRESQL GLOBAL DEVELOPMENT GROUP *PostgreSQL: The world's most advanced open source database* [online]. [citováno 20-04-2014] Dostupné z: http://www.postgresql.org/

[20] MONGODB INC *MongoDB* [online]. [citováno 20-04-2014] Dostupné z: https://www.mongodb.org/

[21] *REST principles explained* [online]. [citováno 20-04-2014] Dostupné z: https://www.servage.net/blog/2013/04/08/rest-principles-explained/

[22] W3C *XMLHttpRequest Level 1* [online]. [citováno 20-04-2014] Dostupné z: http://www.w3.org/TR/XMLHttpRequest/

[23] SENCHA INC *JavaScript Framework for Building Rich Desktop Web Applications* [online]. [citováno 20-04-2014] Dostupné z: http://www.sencha.com/products/extjs/

[24] SASS *Sass: Syntactically Awesome Style Sheets* [online]. [citováno 20-04-2014] Dostupné z: http://sass-lang.com/

[25] W3C *Introduction to Web Components* [online]. [citováno 20-04-2014] Dostupné z: http://www.w3.org/TR/components-intro/

[26] *Redis* [online]. [citováno 20-04-2014] Dostupné z: http://redis.io/

[27] *Protocol Buffers* [online]. [citováno 20-04-2014] Dostupné z: https://code.google.com/p/protobuf/

[28] *Snappy* [online]. [citováno 20-04-2014] Dostupné z: http://code.google.com/p/snappy/

[29] *RabbitMQ* [online]. [citováno 20-04-2014] Dostupné z: https://www.rabbitmq.com/

[30] *AMQP Advanced Message Queuing Protocol* [online]. [citováno 20-04-2014] Dostupné z: http://www.amqp.org/

[31] MICROSOFT CORPORATION *Message Broker* [online]. [citováno 20-04-2014] Dostupné z: http://msdn.microsoft.com/en-us/library/ff648849.aspx

[32] *PHP-FPM - A simple and robust FastCGI Process Manager for PHP* [online]. [citováno 20-04-2014] Dostupné z: http://php-fpm.org/

[33] *Slim Framework* [online]. [citováno 20-04-2014] Dostupné z: http://www.slimframework.com/

[34] PHP GROUP *PHP: New features* [online]. [citováno 20-04-2014] Dostupné z: http://www.php.net/manual/en/migration53.new-features.php

[35] SKVORC, Bruno *Battle of the Autoloaders: PSR-0 vs. PSR-4* [online]. [citováno 20-04-2014] Dostupné z: http://www.sitepoint.com/battle-autoloaders-psr-0-vs-psr-4/

[36] ORACLE *Core J2EE Pattenrs - Front Controller* [online]. [citováno 20-04-2014] Dostupné z: http://www.oracle.com/technetwork/java/frontcontroller-135648.html